ソフトウェア工学 80－6
（１９９１．７．１８）

# 設計情報リポジトリにおける一貫性管理機能

岡 敦子　山本 修一郎　磯田 定宏
ＮＴＴソフトウェア研究所

ソフトウェアの品質および生産性を向上するためには，各種開発支援ツールが生成する設計情報を一元管理することにより有効利用する必要がある．このため，データベースを用いてソフトウェアの設計情報を一元管理することにより，各種の開発支援ツールを統合化する設計情報リポジトリが重要である．設計情報リポジトリ内に管理されている多種多様な設計情報を一元管理するためには，設計情報の追加／削除／更新などの修正に対して，設計情報リポジトリ内の一貫性を保証する一貫性管理機能が必要でる．

本資料では，まず，構造化分析／設計支援ツールが生成する設計情報，プログラム情報などの設計情報間の関係を整理しモデルを作成する．このモデルに基づき設計情報リポジトリの一貫性管理機能を実現するための一貫性維持管理方式を提案する．

# Consistency Management of the Design Information Repository

Atsuko Oka Shuichiro Yamamoto and Sadahiro Isoda

NTT Software Laboratories

NTT Shinagawa TWINS Bldg.,1-9-1 Kohnan, Minato-Ku, Tokyo 108, Japan

To improve software quality and productivity, it is important to make use of various kinds of information concerning the software under development and to integrate the CASE tools which generate this information. The software design information repository consistently manages design information by means of a data base. The repository provides integrated information management which allows various kinds of information and program code to be stored with their inter-related elements.

This paper describes the basic concepts and characteristics of the software design information repository.

## 1．はじめに

ソフトウェアの品質および生産性を向上するためには，①ソフトウェア開発の上流工程から下流工程までを一貫して支援する各種の開発支援ツールを高機能ワークステーション上に構築し，ソフトウェアの分散開発を容易化することと，②これらの開発支援ツールを統合化することにより，ソフトウェアを構成する各種の設計情報を有効利用することが重要である．

このため，データベースを用いてソフトウェアの設計情報を一元管理することにより，各種の開発支援ツールを統合化する設計情報リポジトリが必要である（図1）．設計情報リポジトリ内に管理されている多種多様な設計情報を一元管理するためには，設計情報の追加／削除／更新などの修正に対して，設計情報を矛盾なく更新する機能が必要である．

本稿では，まず，設計情報リポジトリ内に管理する設計情報とその関係を整理することにより設計情報モデルを構成する．さらに，設計情報リポジトリ内の設計情報に対する修正作業時の一貫性を保証するために，修正作業ごとに一貫性を定義するとともに，設計情報モデルに基づく影響探索アルゴリズムを提案する．

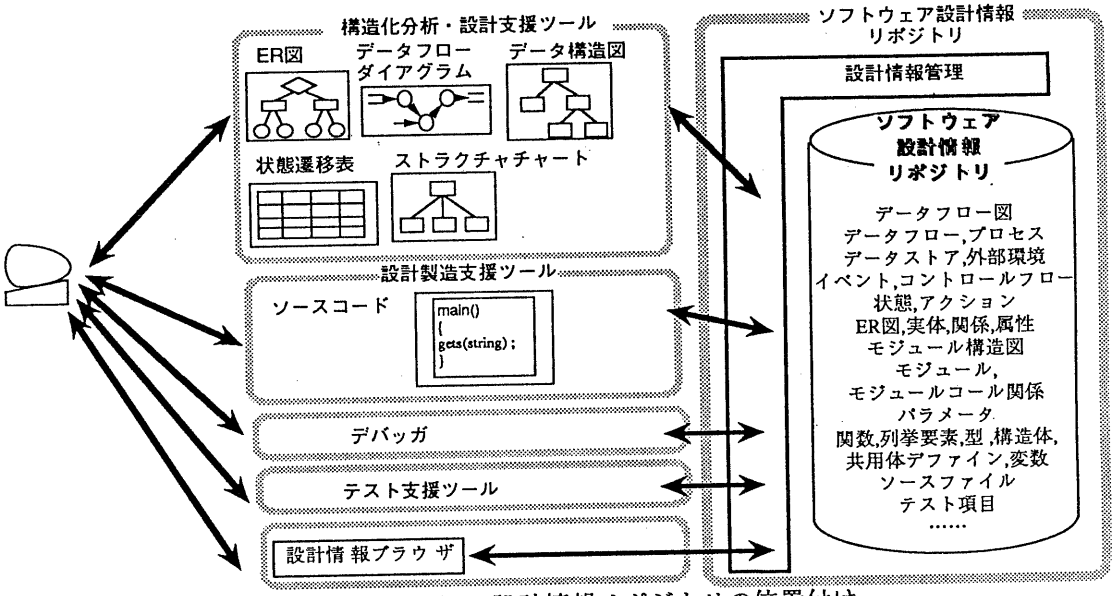



図1 設計情報リポジトリの位置付け

## 2．設計情報モデル

設計情報リポジトリは，ソフトウェアを構成する各種の情報，すなわち，テキスト，図表およびその要素を互いに関連付けた形で記録するためのデータ構造である．設計情報の内容は，要求仕様，ファイル定義，プログラム構成，モジュール仕様，テスト項目など，論理的な設計仕様から物理的なプログラム・コードまで多様な抽象度を持つ情報から構成されている．しかも，それらの間の関係が複雑である．たとえば，データフロー図上のデータフローは名前を媒介としてデータ構造図上のデータ定義に関係づけられている．さらに，モジュール構造図やソースコード上のパラメータにまで関係がおよぶ．

設計情報リポジトリでは，多様かつ複雑なソフトウェアに関するさまざまの情報を管理するため，以下の手順で設計情報モデルを構成した．

①管理情報項目の抽出
②管理情報項目間の関係の分類
③設計情報モデルの構築

### （1）設計情報リポジトリにおける管理情報項目の抽出

ソフトウェアの設計情報として各工程生産物を管理する必要がある．設計情報リポジトリの管理情報例を表1に示す．

表1　設計情報リポジトリの管理情報（例）

| 工程 | 工程生産物 | 構成要素 |
|---|---|---|
| 基本設計 | データフローダイアグラム | データフロー，プロセス，データストア<br>外部環境 |
| | ＥＲダイアグラム | 実体，関係，属性 |
| 詳細設計 | データ構造ダイアグラム | データ構造，データ項目，データの構造関係 |
| | ストラクチャチャート | モジュール，パラメータ，制御情報，<br>モジュールコール関係 |
| | テーブル設計書 | データ構造，データ項目，データの構造関係 |
| | メッセージ仕様書 | メッセージ |
| 製造 | ソースファイル | 関数，関数コール関係，変数，構造体，共用体<br>列挙要素，デファイン，パラメータ |
| | ヘッダファイル | 変数，構造体，共用体，デファイン，列挙要素 |
| デバグ | テスト仕様書 | テスト項目 |

（2）設計情報リポジトリの管理情報項目間の関係の分類

ソフトウェアの設計情報はそれ以上分割できない構成要素とそれらを要素とする複合オブジェクト（工程生産物）に分けることができる．したがって，設計情報間のすべての関係は，(1)複合オブジェクト間の関係　(2)複合オブジェクトと構成要素間の関係　(3)構成要素間の関係　の３種類のいずれかに分類できる．さらに(2)および(3)の関係は，以下のように分類できる．

(2)複合オブジェクトと構成要素間の関係…複合オブジェクトの構成関係，構成要素の詳細化関係

(3)構成要素間の関係…設計情報の定義と参照に着目した関係

（定義－定義関係）　①結合関係　②類似関係　③定義実現関係　④定義一致関係

（定義－参照関係）　⑤強定義参照関係　⑥弱定義参照関係

（参照－参照関係）　⑦参照実現関係　⑧参照独立関係

設計情報Ｙを定義するために設計情報Ｘの定義が必要となるとき，ＸをＹの定義側という．このとき，ＹをＸの参照側という．たとえば，定義参照関係にある場合の定義側設計情報，構成関係での複合オブジェクト，詳細化関係での詳細化対象となる構成要素が定義側設計情報である．以下では，定義参照関係について次の前提を置く．すなわち，ＸがＹの定義側かつＹがＺの定義側であるとき，ＺがＸの定義側になることはない．

構成要素間の関係は，定義参照関係（⑤，⑥）だけでなく，定義－定義関係（①，②，③，④），参照－参照関係（⑦，⑧）が存在する．

これらの設計情報に関する関係の具体例を図２に示す．

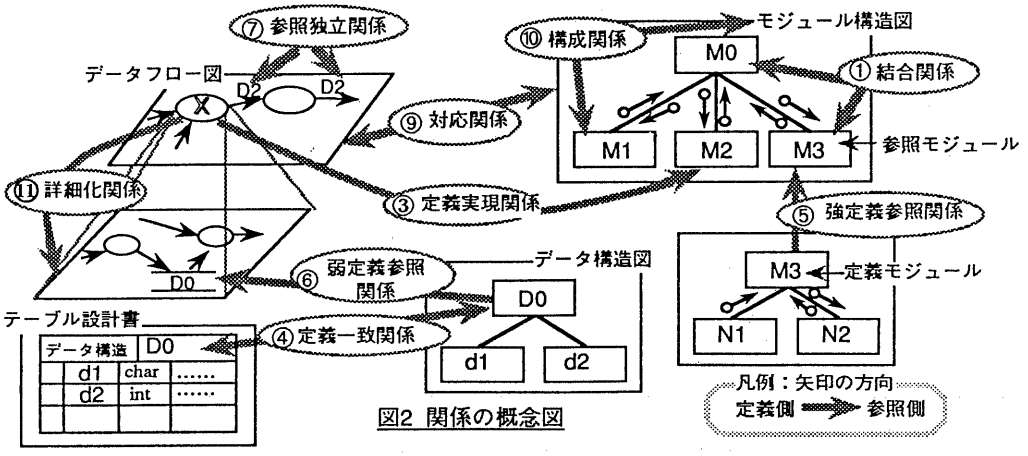



図2 関係の概念図

このように設計情報の構造，ならびに定義，参照の仕方に着目することによりすべての関係を整理できる（表２）．ただし，複合オブジェクト間の定義参照関係は対象としていない．以下では，構成

要素間の定義参照関係に限定して議論を進める．構成要素間の関係の分類を図３に示す．

表２　設計情報間の関係の分類

| 関係名 | | 説明 | 設計情報の構造 | 関係の具体例 | ＊番号 |
|---|---|---|---|---|---|
| 対応関係 | | 複合ｵﾌﾞｼﾞｪｸﾄ間の対応関係 | 複合ー複合 | ﾃﾞｰﾀ構造図とﾍｯﾀﾞﾌｧｲﾙ<br>ﾃﾞｰﾀﾌﾛｰ図とｽﾄﾗｸﾁｬﾁｬｰﾄ | ⑨ |
| 定義ー参照 | 構成関係 | 複合ｵﾌﾞｼﾞｪｸﾄとその内部の構成要素との間の関係 | 複合ー構成要素 | ﾃﾞｰﾀﾌﾛｰﾀﾞｲｱｸﾞﾗﾑと<br>ﾌﾟﾛｾｽ／外部環境／<br>ﾃﾞｰﾀｽﾄｱ／ﾃﾞｰﾀﾌﾛｰ | ⑩ |
| | 詳細化関係 | 構成要素と構成要素の内部を詳細に説明する複合ｵﾌﾞｼﾞｪｸﾄ | | ﾌﾟﾛｾｽと詳細化したﾃﾞｰﾀﾌﾛｰﾀﾞｲｱｸﾞﾗﾑ | ⑪ |
| 定義ー定義 | 結合関係 | 他の構成要素同士の接続関係となっている構成要素 | 構成要素ー構成要素 | ﾓｼﾞｭｰﾙ間のｺｰﾙ関係<br>ﾃﾞｰﾀﾌﾛｰ関係 | ① |
| | 類似関係 | ある複合ｵﾌﾞｼﾞｪｸﾄで定義された構成要素と同種の複合ｵﾌﾞｼﾞｪｸﾄ内の他の構成要素が類似している関係 | | 類似する処理を行う<br>ﾓｼﾞｭｰﾙ同士 | ② |
| | 定義実現関係 | 上流の複合ｵﾌﾞｼﾞｪｸﾄ上の構成要素を下流の複合ｵﾌﾞｼﾞｪｸﾄの構成要素が実現する関係 | | ﾌﾟﾛｾｽと実現ﾓｼﾞｭｰﾙ間の関係 | ③ |
| | 定義一致関係 | ある構成要素が複数種の複合ｵﾌﾞｼﾞｪｸﾄにより定義される関係．すなわち，同一の構成要素を異なるﾋﾞｭｰで定義している． | | ﾃﾞｰﾀ構造図上のﾃﾞｰﾀと<br>ﾃｰﾌﾞﾙ設計書上のﾃﾞｰﾀ | ④ |
| 定義ー参照 | 強定義参照関係 | ある複合ｵﾌﾞｼﾞｪｸﾄで定義された構成要素を同種の複合ｵﾌﾞｼﾞｪｸﾄ内の他の構成要素が参照する関係 | | 定義ﾓｼﾞｭｰﾙと<br>参照ﾓｼﾞｭｰﾙ | ⑤ |
| | 弱定義参照関係 | ある複合ｵﾌﾞｼﾞｪｸﾄで定義された構成要素を異なる種類の複合ｵﾌﾞｼﾞｪｸﾄ上の構成要素が参照する関係 | | ﾃﾞｰﾀ構造図上のﾃﾞｰﾀと<br>ﾃﾞｰﾀﾌﾛ図上のﾃﾞｰﾀﾌﾛｰ | ⑥ |
| 参照ー参照 | 参照独立関係 | 他の複合ｵﾌﾞｼﾞｪｸﾄで定義した構成要素を参照している構成要素間の関係 | | ﾃﾞｰﾀﾌﾛｰ図上の<br>ﾃﾞｰﾀﾌﾛｰ同士 | ⑦ |
| | 参照実現関係 | 他の複合ｵﾌﾞｼﾞｪｸﾄで定義した構成要素を参照する異なる複合ｵﾌﾞｼﾞｪｸﾄの構成要素間の実現関係 | | ﾃﾞｰﾀﾌﾛｰ図上のﾃﾞｰﾀﾌﾛｰと<br>ﾓｼﾞｭｰﾙ構造図上のﾊﾟﾗﾒｰﾀ | ⑧ |

（注）　＊：番号　…図２および図３での対応する番号

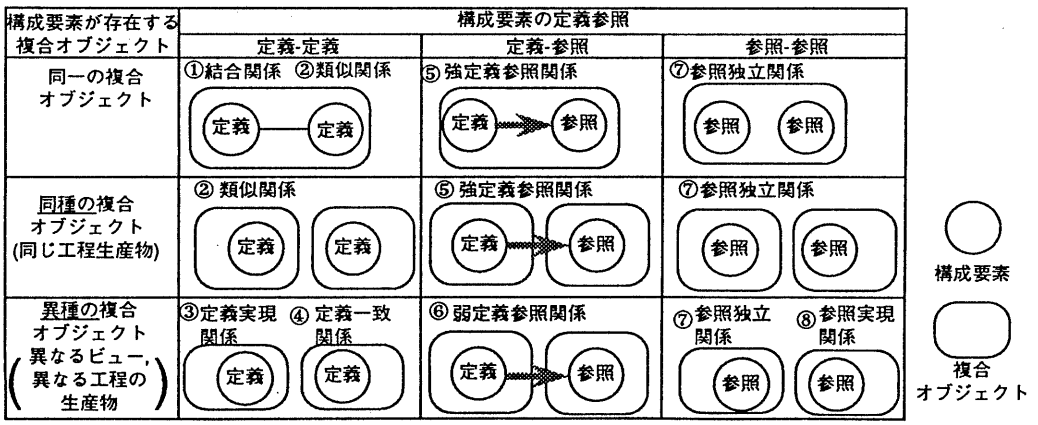



図３　構成要素間の関係の分類

（３）設計情報モデルの構築

表１に示した構成要素および工程生産物（複合オブジェクト）をエンティティ（実体）とし，表２で示したエンティティ間の関係を設計情報間の関係として整理することにより，設計情報をＥＲモデルによりモデル化できる．

## ３．設計情報の一貫性管理

### ３．１　設計情報の一貫性

設計情報リポジトリ内の多種多様な設計情報を一元管理するためには，設計情報の追加／削除／修正といった開発中の作業に対して，一貫性を保証する必要がある．

設計情報リポジトリの一貫性保証機能は，大きく以下の２種類に分けられる．

①設計情報の一貫性検証　　…任意の時点において，設計情報が一貫しているか検証する．

②修正作業時の一貫性保証　　…一貫性を保証しながら修正を行う．

以下では，②修正時の一貫性について保証するための更新処理方式について述べる．

（１）設計情報の一貫性規則

設計情報の一貫性を以下の３種類に分類した．

［規則１：定義内容の一貫性］

同一名称に対して異なる定義を与えない（二重定義の禁止）．

［規則２：定義／参照関係の一貫性］

- 定義名の参照が存在する．
- 未定義名の参照が存在しない．

［規則３：参照間の一貫性］

定義名に対する複数の参照間で成立すべき条件

［データフローバランス：データフロー図における参照データ間の一貫性規則（図４）］

プロセスＰ１に入出力するデータフローと，Ｐ１を詳細化したデータフロー図Ｑの境界に入出力するデータフローとが対応する．このとき，Ｐ１に入出力するデータフローＤ０に対して，Ｑの境界に存在するデータフローは以下のいずれかの条件を満たさなければならない．

条件１：データフローＤ０がデータフロー図Ｑ上に存在する．

条件２：データ構造図Ｓ上に定義されたＤ０のすべての子データｄｉ(i=1,…,n)がデータフロー図Ｑ上に存在する．

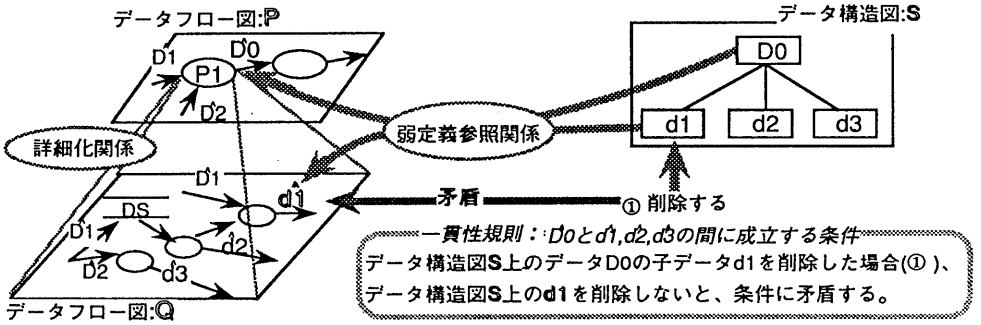



図4　一貫性規則例-データフローのバランス-

（２）設計情報に対する修正作業の分類

設計情報リポジトリ内の設計情報に対する修正作業は，①設計情報の削除　②設計情報の追加　③設計情報の更新－名標の修正－　④設計情報の更新－属性の修正－　の４種類に分類できる．

③，④以外の更新作業は，すべて設計情報の削除／追加（①，②）に帰着できる．

### ３．２　一貫性の保証方式

設計情報リポジトリの一貫性を保証するために，修正作業ごとに保証すべき一貫性を定義し，修正作

業時の一貫性維持管理方式を明らかにする．

（1）一貫性の定義

修正作業時に，必ずしもすべての一貫性規則を保証しなくてもよい．例えば，開発作業ではデータについての定義だけ先に行ったため，未参照となる場合がよくある．これは，規則2：定義参照関係の一貫性規則に違反するが，データの参照箇所のチェックツールによって検出すればよい．すなわち，本論文では，以下の一貫性規則を保証することを前提として議論を進める．

修正作業時の一貫性：修正作業において，一貫性規則1，3を保証する．
一貫性規則2については，チェックツールによって検出する方式を採用する．

（2）一貫性の維持管理方式の定義

設計情報リポジトリ内の設計情報は種類が多く，関係も複雑である．わかりやすく効率的に更新処理を定義するためには，2．で示した設計情報モデルに基づき，更新処理を明確化する必要がある．

設計情報リポジトリへの修正作業の前後において一貫性を保つためには，設計情報ごとに関係をたどり，修正を伝播させる必要がある．すなわち，修正作業ごとに修正対象や関係種別に従って，（1）で述べた一貫性を保証する更新処理を実行する．

各関係について，定義側設計情報を修正した場合の参照側設計情報に対する更新マトリックス MD，参照側設計情報を修正した場合の定義側設計情報に対する更新マトリックス MR をそれぞれ定義する（表3）．更新マトリックスの行は関係種別を，列は修正作業を示す．

定義一致関係のように定義－定義関係の場合にはMD，参照独立関係のように参照－参照関係の場合はMRを用いる．

影響波及先への操作は，MD（R，Opr），MR（R，Opr）（R：関係種別，Opr：修正作業）によって示すことができる．また，修正の可／不可を値の正負により示し，一貫性規則のチェックの要／不要を絶対値によって示す．すなわち，更新マトリックスの各欄の値が示す意味は以下の通りである．

①2の場合：参照間の一貫性規則が成立する場合，影響波及を行う．
成立しない場合，影響波及を行わない．
②1の場合：定義側と同様に削除／追加／名標の修正／属性の修正を行う
③0の場合：削除の場合…関係のみ削除する　その他の場合…何もしない
④-1の場合：修正は不可
⑤-2の場合：参照間の一貫性規則が成立する場合，修正を行わない．
成立しない場合，修正を行う．

表3　設計情報を修正した場合の更新処理定義ﾏﾄﾘｯｸｽ（MD｜MR）

| 関係種別 | | 定義側修正後の参照側の更新処理定義ﾏﾄﾘｯｸｽ MD | | | | 参照側修正後の定義側更新処理定義ﾏﾄﾘｯｸｽ MR | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 定義側削除 | 定義側追加 | 定義側更新 名標 | 定義側更新 属性 | 参照側削除 | 参照側追加 | 参照側更新 名標 | 参照側更新 属性 |
| Ⅰ | 対応関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅱ | 構成関係 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅲ | 詳細化関係 | 1 | 2 | 1 | 1 | −2 | −2 | −1 | −1 |
| Ⅳ | 結合関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅴ | 類似関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅵ | 定義実現関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅶ | 定義一致関係 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Ⅷ | 強定義参照関係 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Ⅸ | 弱定義参照関係 | 2 | 2 | 0 | 1 | 0 | 0 | 0 | 1 |
| Ⅹ | 参照独立関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Ⅺ | 参照実現関係 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

たとえば，ある複合オブジェクトXの定義を削除した場合についての更新処理は以下の通りである．複合オブジェクトXから構成要素Aへの影響波及は，MD（構成関係，削除）＝1，すなわち，構成要素Aを削除する．さらに，構成要素Aと関連する設計情報Qを検出して，Qに対してこの操作を反復的に

適用する．この結果，関連する設計情報がない，または，MD（R，Opr）＝０となるまで処理を実行する．

（３）修正時の影響探索アルゴリズム

（２）の例に示したように，表３に示した更新マトリックスを適用することにより，設計情報リポジトリ内に修正内容が伝播していく．影響探索アルゴリズムを図５に示す．ここでは，以下の記号を用いる．設計情報Ｘに対する影響範囲探索アルゴリズムをＦ(Ｘ)，Ｘに対して行う修正作業をOpr，設計情報ａ，ｂ間の関係をＲ(ａ，ｂ)，関係Ｒの種類をＴ(Ｒ)，設計情報Ｘ，Ｙの間に成立している一貫性規則をＩ(Ｘ，Ｙ)，更新マトリックスを　MD，MRとして表す．

モジュール構造図（複合オブジェクト）を削除する場合に本アルゴリズムを適用した例を付図１に示す．この場合，削除したモジュール構造図上の構成要素，および，削除した構成要素と他の設計情報との関連が削除される．しかし，関連先の他の複合オブジェクト，他の複合オブジェクト上の構成要素は削除されず，設計情報リポジトリ内に残る．

*F(X)*

Step1: {Xが修正できない場合をチェックする}
Xが参照側となるすべての関係Rに対し、R(D,X)を満たすすべてのDについて以下の処理を繰り返す。
begin
MR(T(R),Opr)= -2のとき{一貫性規則をチェックする場合}
Xに対してあるYが存在し,一貫性規則 I(X,Y)が成立しているとき,Xに対する修正は行わない。
goto Step5
MR(T(R),Opr)= -1のとき、goto Step5
end

Step2: Xが参照側となるすべての関係Rに対し、R(D,X)を満たすすべてのDについて以下の処理を繰り返す。
begin
MR(T(R),Opr)=1のときに限り、Dに対してF(D)を実行する。
Opr=削除のときに限り、　　関係R(D,X)を削除する。
end

Step3: Xが定義側となるすべての関係Rに対し、R(X,U)を満たすすべてのUについて以下の処理を繰り返す。
begin
MR(T(R),Opr)=2 とき、{一貫性規則をチェックする場合}
Uに対してあるYが存在し,一貫性規則 I(U,Y)が成立しているとき,Uに対してF(U)を実行する。
MR(T(R),Opr)=1のときに限り、Uに対してF(U)を実行する。
Opr=削除のときに限り、　関係R(X,U)を削除する。
end

Step4: Xに対してOprを実行する。

Step5: 終了

図5　影響探索アルゴリズム

## ４．考察

（１）保証方式の柔軟性

一貫性の保証方法を考えた場合，①常にすべての一貫性規則を満たすことを要求する強一貫性保証と②一貫性規則を満たさない部分を更新処理とは別にチェックツールにより検出する弱一貫性保証がある．設計情報リポジトリは，要求される一貫性保証の強度に対応する必要がある．更新マトリックス方式では以下のように値を持つ更新マトリックスを個別に用意することによって両方式に柔軟に対応できる．定義側設計情報を削除した場合の例を以下に示す．

①強一貫性保証：MD（弱定義参照関係，削除）＝MD（強定義参照関係，削除）＝１
参照側まで削除することにより，常に一貫性を保証する．

②弱一貫性保証：MD（弱定義参照関係，削除）＝MD（強定義参照関係，削除）＝０
参照側は削除しないが，別起動のチェックツールによって未定義情報を検出する．

（２）関係への修正作業

表３では関係を定義する設計情報を修正対象として，更新マトリックスを定義した．しかし，データフロー，データの親子関係のような結合関係もまた複合オブジェクトの構成要素であり，修正対象となり得る．これらは関係自身が名標を持つかどうかにより，以下のように更新マトリックスを適用するこ

とができる．

①関係自身が名標を持つ（データフローなど）

他の設計情報と同様にアルゴリズムを適用する．

②関係自身が名標を持たない（データの親子関係，モジュールコール関係）

以下の置き換え規則により，更新マトリックスを適用する．

関係の追加：関係先設計情報を追加登録する作業に置き換える．

関係の削除：関係先の設計情報を削除する作業に置き換える．

## 5．おわりに

本資料では，設計情報間の関係を１１種類に分類し，この関係に基づいて，設計情報リポジトリの一貫性を保証する機能として，一貫性の定義，および，影響探索アルゴリズムを提案した．今後は，設計情報リポジトリ内の設計情報の一貫性を検証する機能について検討を進める予定である．

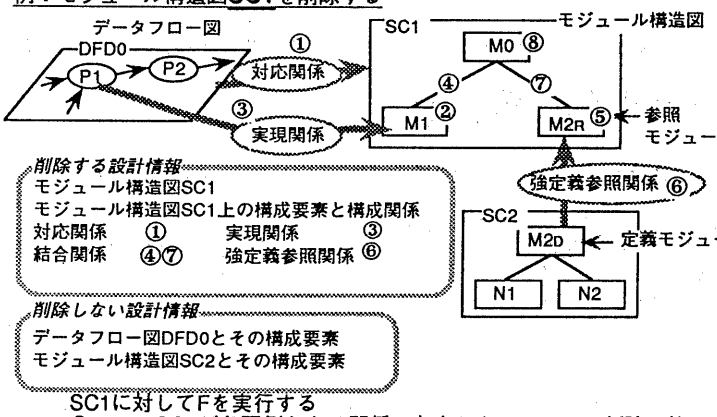
【参考文献】

［１］岡,山本,"設計情報リポジトリを用いた改造支援方法",情報処理学会第41回全国大会,(1990).

［２］鰺坂,沢田,山下,松本,"ｿﾌﾄｳｪｱｴﾝｼﾞﾆｱﾘﾝｸﾞ・ﾃﾞｰﾀﾍﾞｰｽKyotoDBのｵﾌﾞｼﾞｪｸﾄﾓﾃﾞﾙ",ﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑ研究会79-4（1990）．

［３］松本,"CASE環境の基礎技術",情報処理第31巻第8号（CASE環境特集）,pp.1020～1027（1990）．

例：モジュール構造図**SC1**を削除する




定義-参照関係の一覧表

| 関係種別 | 定義側 | 参照側 |
|---|---|---|
| 構成関係 | SC1 | M0,M1,M2R |
| 構成関係 | SC2 | M2D,N1,N2 |
| 構成関係 | DFD0 | P1,P2 |
| 強定義参照関係 | M2D | M2R |

定義-定義関係の一覧表

| 関係種別 | 定義側 | 定義側 |
|---|---|---|
| 対応関係 | SC1 | DFD0 |
| 実現関係 | M1 | P1 |
| 結合関係 | M0 | M1,M2R |
| 結合関係 | M2D | N1,N2 |

SC1に対してFを実行する

*Step1,2* SC1が参照側となる関係は存在しない…SC1は削除可能

*Step3* R(SC1,DFD0)についてMR(対応関係,削除)=0…対応関係R(SC1,DFD0)を削除する(①)

R(SC1,M1)についてMD(構成関係,削除)=1

…M1に対してFを実行する

*Step1,2* M1が参照側となる関係は存在しない…M1は削除可能

*Step3* R(M1,P1)について MR(実現関係,削除)=0…実現関係R(M1,P1)を削除する(③)。

R(M1,M0)について MR(結合関係,削除)=0…結合関係R(M1,M0)を削除する(④)。

*Step4* M1を削除する(②)。

*Step5* M1に対する処理は終了。

関係R(SC1,M1)を削除する。

R(SC1,$M2_R$)についてMD(構成関係,削除)=1

…$M2_R$に対してFを実行する

*Step1* R($M2_D$,$M2_R$)についてMR(強定義参照関係,削除)=0…$M2_R$は削除可能

*Step2* 強定義参照関係R($M2_D$,$M2_R$)を削除する(⑥)。

*Step3* R($M2_R$,M0)について MR(結合関係,削除)=0…結合関係R($M2_R$,M0)を削除する(⑦)。

*Step4* $M2_R$を削除する(⑤)。

*Step5* $M2_R$に対する処理は終了。

関係R(SC1,$M2_R$)を削除する。

R(SC1,M0)についても同様…F(M0)を行う。R(SC1,M0),M0を削除する(⑧)。

*Step4* SC1を削除する。

*Step5* 終了

付図1 影響探索アルゴリズムの適用